VAX-11 BASIC

Pocket Guide

digital

DIGITAL EQUIPMENT CORPORATION

Printed in U.S.A.

digital software

VAX–11 BASIC
Pocket Guide

Order No. AV–M433A–TE

February 1982

This guide contains useful information about VAX–11 BASIC.

OPERATING SYSTEM AND VERSION: VMS

SOFTWARE VERSION: VAX–11 BASIC V1

digital equipment corporation, maynard, massachusetts

The information in this document is subject to change without notice and should not be construed as a commitment by Digital Equipment Corporation. Digital Equipment Corporation assumes no responsibility for any errors that may appear in this document.

The software described in this document is furnished under a license, and may be used or copied only in accordance with the terms of such license.

No responsibility is assumed for the use or reliability of software on equipment that is not supplied by DIGITAL or its affiliated companies.

Copyright © 1982 Digital Equipment Corporation

The following are trademarks of Digital Equipment Corporation:

DEC	VT	IAS
DECUS	DECsystem–10	MASSBUS
DECnet	DECSYSTEM 20	PDT
PDP	DECwriter	RSTS
UNIBUS	DIBOL	RSX
VAX	EduSystem	VMS

Contents

	Page
Meta-Language Elements and Conventions	1
Commands	4
Statements	8
Functions	26
Operators	34
Operator Precedence	35
Sample Program Coding Template	36
Principles of Transportability	44
ASCII Character Codes	48
Predefined Constants	60
VAX–11 BASIC Reserved Words	62

Meta-Language Elements and Conventions

[ ] (brackets)	enclose an optional portion of a general format. When they enclose vertically stacked entries, brackets indicate that you can select one of the enclosed elements.
{ } (braces)	enclose a mandatory portion of a general format. When they enclose vertically stacked entries, braces indicate that you must choose one of the enclosed elements.
... (ellipsis)	indicates an option of repeating the immediately preceding element.
exp	is an expression.
str-exp	is an expression whose value is a string.
num-exp	is an expression whose value is a number.
int-exp	is an expression whose value is an integer.
chnl-exp	is an I/O channel number between 0 and 99.
cond-exp	is either a logical or relational expression.
subs-exp	is a subscripted expression.

vbl	is a simple variable.
num-vbl	is a numeric variable.
str-vbl	is a string variable.
cnst	is a constant.
int-cnst	is an integer constant.
num-cnst	is a whole number. A percent sign is optional.
str-cnst	is a string constant.
lin-num	is a line number. It is a restricted form of <const> that 1) must be numeric, 2) must not contain "." or "E", 3) must be greater than zero, and 4) must be less than 32768.
symb	is a symbol.
arr-nam	is the name of an array.
cnst-nam	is the name of a constant.

func-nam	is the name of a function.
map-nam	conforms to restrictions for naming real variables and is global to the program.
d-type	specifies one of the following data types: REAL, SINGLE or DOUBLE; INTEGER, WORD or LONG; or STRING.
num-arr	is a numeric array.
FILL–item	is used in place of any variable before the first occurrence of <d-type>. After the first occurrence of <d-type>, use only FILL for a <FILL–item>.
stmnts	is a statement.

Commands

APPEND	**file-spec**	Merges a source file with the program in memory.
COMPILE	**[/qualifier...] [file-name]**	Compiles the program in memory.
	/[NO]BOUNDS	Checks the range of array elements.
	/[NO]CROSS_REFERENCE	Requests cross-reference listing.
	/[NO]DEBUG	Defines local symbols for debugger.
	/DOUBLE	Specifies 64–bit floating point length.
	/[NO]LINES	Enables line number error reporting.
	/[NO]LIST	Creates a program listing file.
	/LONG	Specifies 32–bit integer length.
	/[NO]MACHINE_CODE	Includes assembler code listing.
	/[NO]OBJECT	Creates linkable object file.
	/[NO]OVERFLOW	Enables integer overflow detection.
	/[NO]SETUP	Disables error checking and initialization routines, and increases execution speed.
	/SINGLE	Specifies 32–bit floating point length.
	/[NO]TRACEBACK	Allows virtual address translation for the debugger and error reporter.
	/WORD	Specifies 16–bit integer length.
CONTINUE		Resumes program execution.
DELETE	**lin-num [,lin-num] [...]** **lin-num – lin-num**	Deletes specified line number(s) or line number range.
EDIT	**[lin-num] [/sr-str/[rp-str]]** **[/occurrence[,sub-line]]**	Replaces the search string with the replacement string, or deletes the search string if no replacement string is specified.
EXIT		Clears memory and returns to DCL command level. CTRL/Z also performs this function.
HELP	**[keyword] [keyword...]**	Displays information about BASIC.
IDENTIFY		Displays compiler name and version.
INQUIRE		Displays information about BASIC and is identical to the HELP command.
LIST[NH]	**[lin-num [,lin-num...]]** **[lin-num – lin-num]**	Displays all lines, specified line number(s), or a line number range.
LOAD	**file-spec [+file-spec...]**	Loads specified object file(s). (See also RUN.)
LOCK	**/qualifier**	Sets defaults for COMPILE command qualifiers and is identical to the SET command.

NEW	**program-name**	Clears memory and assigns a program name.
OLD	**file-spec**	Brings a source file into memory.
RENAME	**program-name**	Assigns a new name to the program in memory.
REPLACE	**[file-spec]**	Writes the program in memory to the specified file (creates a new version).
RUN[NH]	**[file-spec]**	Executes the program in memory and any object modules brought into memory by LOAD, or the specified source program.
SAVE	**[file-spec]**	Writes the program in memory to the specified file. (See also REPLACE, UNSAVE.)
SCALE	**n**	Controls accumulated round-off errors and is used with double precision.
SCRATCH		Clears memory, resets the program name, and removes any loaded files.
SEQUENCE	**[start-line-num] [,increment]**	Generates line numbers for text input. Exit by pressing CTRL/Z.
SET	**/qualifier**	Sets defaults for COMPILE command qualifiers.
SHOW		Displays default compile command qualifiers. (See also LOCK, SET.)
UNSAVE	**[file-spec]**	Deletes the specified file. (See also SAVE.)

Statements

CALL sub-name [BY REF | BY DESC | BY VALUE] (exp [BY REF | BY DESC | BY VALUE] [exp [BY REF | BY DESC | BY VALUE] ...])

Transfers control to a subprogram, optionally passes it parameters, and stores the calling program's return location. Using BY REF, BY DESC, or BY VALUE before the left parenthesis defines parameter passing defaults for the following parameters, thus eliminating the defaults otherwise determined by each parameter's data type.

CHAIN str-exp
Transfers control to a different executable BASIC program.

CHANGE str-exp TO num-array
Converts a string of characters to a numeric array containing each character's decimal ASCII value. Array element 0 returns the string's length.

CHANGE num-array TO str-vbl
Converts num-array to a string of ASCII characters. Array element 0 must specify the string's length.

CLOSE #chnl-exp [,chnl-exp]...
Ends access to a device or file.

COM[MON] [(com-nam)] [d-type] { num-vbl | str-vbl[=int-exp] | num-arr(num-cnst[,num-cnst]) | str-arr(num-cnst[,num-cnst])[=num-cnst] | FILL–item } [...] [...] [...] [...] [...]

Defines a static-length, shared storage area, called a COMMON block, that a program and its subprograms can access by name.

DATA { num-cnst | str-cnst | unquoted-str | quoted-str } [, ...]

Creates a data block for the READ statement.

DECLARE d-type vbl [, vbl] ... [,[d-type] vbl]...
Explicitly assigns a data type to a variable.

DECLARE d-type CONSTANT cnst-nam = value [,cnst-nam = value...]
Explicitly assigns a data type to a symbolic constant and defines its value.

DECLARE d-type FUNCTION func-nam [,func-nam...]
Explicitly assigns a data type to a function.

DEF FNvbl1 [([d-type] vbl2 [,[d-type] vbl3] ...)] = exp
Defines a single-line, user-defined function.

DEF FNvbl1 [([d-type] vbl2 [,[d-type] vbl3] ...)]
Defines a multi-line, user-defined function. (See also FNEND.)

DELETE # chnl-exp
Removes a record from a file.

DIM[ENSION] d-type arr-nam(int-cnst [,int-cnst]) [,arr-nam...]
Declares an array in memory and specifies the number of dimensions and the maximum value of each subscript.

DIM[ENSION] [#]chnl-exp,[d-type] arr-nam(int-cnst[,int-cnst])[=int-cnst] [...]
Declares a virtual array; specifies the number of dimensions and the maximum value of each subscript; specifies string length for virtual string arrays.

END
Marks the end of a program, closes all open files, and releases all program storage.

EXTERNAL d-type [{ **FUNCTION** / **CONSTANT** }] **symb [,symb ...]**

Enables access to external program symbols, system services, and function subprograms.

FIND #chnl-exp
Locates the next logical record in a sequential, relative, or indexed file.

FIND #chnl-exp ,RECORD int-exp
Locates the specified record in a relative or virtual file.

FIND #chnl-exp, KEY #int-exp [{ **EQ** / **GE** / **GT** } { **str-exp** / **int-exp** }]

Locates the specified record in an indexed file.

FNEND
Ends a multi-line function definition. (See also multi-line DEF.)

FNEXIT
Exits from a multi-line function. Equivalent to an unconditional branch to FNEND.

FOR num-vbl = num-exp1 TO num-exp2 [STEP num-exp3]
Begins a program loop and defines loop execution. (See also NEXT.)

FOR num-vbl = num-exp1 [STEP num-exp2] { **WHILE** / **UNTIL** } **exp**

Begins a program loop and defines loop execution under a WHILE or UNTIL condition. (See also NEXT.)

stmnt FOR num-vbl = num-exp1 TO num-exp2 [STEP num-exp3]
Executes the preceding statement and defines the conditions for its repetitive execution.

FREE #chnl-exp
Unlocks all records or buckets for a channel.

FUNCTION d-type [([d-type] { variable / array [BY REF] } [....])]

Begins a function subprogram and specifies its parameters.

FUNCTIONEND
Ends a function subprogram and returns control to the invoking program.

FUNCTIONEXIT
Returns control to the invoking program. Equivalent to an unconditional branch to FUNCTIONEND.

GET #chnl-exp
Reads the next logical record in a sequential, relative, or indexed file.

GET #chnl-exp,RECORD int-exp
Reads the specified record in a relative or virtual file.

GET #chnl-exp, KEY #int-exp [{ GE / EQ / GT } { str-exp / int-exp }]

Reads the specified record in an indexed file.

{ **GOSUB** / **GO SUB** } **lin-num**

Transfers control to lin-num and stores the return address.

{ **GOTO** / **GO TO** } **lin-num**

Transfers control to lin-num.

IF cond-exp THEN { stmnt(s) / lin-num } [**ELSE** { stmnt(s) / lin-num }]

Evaluates cond-exp and, if true, transfers control to lin-num or executes the statement(s) after THEN. If false, transfers control to lin-num or the statement(s) after ELSE.

stmnt IF cond-exp
Evaluates cond-exp and executes the preceding statement only if cond-exp is true.

INPUT [#chnl-exp,] [**str-cnst** { , ; }] **vbl [,vbl]...**

Assigns a value from a terminal or from a terminal-format file to each program variable.

INPUT LINE [#chnl-exp,] [**str-cnst** { , ; }] **str-vbl**

Assigns str-vbl a string value, including line terminator(s), from a terminal or from a terminal-format file.

KILL str-exp
Deletes the file specified by str-exp.

[LET] vbl [,vbl]... = exp
Assigns a value to each variable.

LINPUT [#chnl-exp,] [**str-cnst** { , ; }] **str-vbl**

Assigns str-vbl a string value from a terminal or from a terminal-format file.

LSET str-vbl [,str-vbl]... = str-exp
Assigns left-justified data to each string variable (does not change string variable length).

MAP (map-nam) [d-type] { **num-vbl** / **str-vbl[=int-exp]** / **num-arr(num-cnst[,num-cnst])** / **str-arr(num-cnst[,num-cnst])[=num-cnst]** / **FILL-item** } [,.... ,.... ,.... ,.... ,....]

Defines a named, shared storage area of static length like the COMMON statement. (See also OPEN.)

MARGIN [#chnl-exp,] int-exp
Specifies the margin width for records in a terminal-format file.

MAT arr-nam = { **CON** / **IDN** / **NUL$** / **ZER** } **[(subs-exp1 [,subs-exp2])]**

Creates an array or redimensions an existing array, and assigns values to array elements.

MAT arr-nam1 = arr-nam2 { + / − / * } **arr-nam3**

Sets arr-nam1 equal to either arr-nam2 or to the matrix sum, difference, or product of arr-nam2 and arr-nam3.

MAT arr-nam1 = (num-exp) * arr-nam2
Multiplies each element of arr-nam2 by num-exp and stores the result in arr-nam1.

MAT arr-nam1 = { **INV** / **TRN** } **(arr-nam2)**

Sets arr-nam1 equal to the inverse or the transposition of arr-nam2.

MAT INPUT [#chnl-exp,] arr-nam [(subs-exp1[,subs-exp2])] [,arr-nam]...
Assigns each arr-nam element a value from a terminal or from a terminal-format file.

MAT LINPUT [#chnl-exp,] str-arr-nam [(subs-exp1[,subs-exp2])]
Assigns each str-arr-nam element a string value from a terminal or from a terminal-format file.

MAT PRINT [#chnl-exp,] arr-nam [(subs-exp1[,subs-exp2])] { **,** / **;** } **[arr-nam...]**

Either prints the array element values at a terminal or assigns each array element's value to a record in a terminal-format file.

MAT READ arr-nam [(subs-exp1 [,subs-exp2])] [,arr-nam...]
Assigns values from DATA statements to array elements.

MOVE { **FROM** / **TO** } **#chnl-exp** { **num-vbl** / **str-vbl [=int-exp1]** / **num-arr-nam (subs-exp1[,subs-exp2])** / **str-arr-nam (subs-exp1[,subs-exp2])[= int-exp2]** / **FILL-item** } [**,...**]

Moves data in a record buffer to or from specified variables.

NAME str-exp1 AS str-exp2
Changes the name of the specified file.

NEXT num-vbl
Ends a FOR loop.

NEXT
Ends an UNTIL or WHILE loop.

NOMARGIN [#chnl-exp]
Cancels the effect of MARGIN.

{ **ON ERROR** / **ONERROR** } { **GO BACK** / **GOBACK** }

Transfers control to the calling program if an error occurs in a DEF or subprogram.

{ON ERROR / ONERROR} {GO TO / GOTO} lin-num

Transfers control to lin-num in the current program if an error occurs.

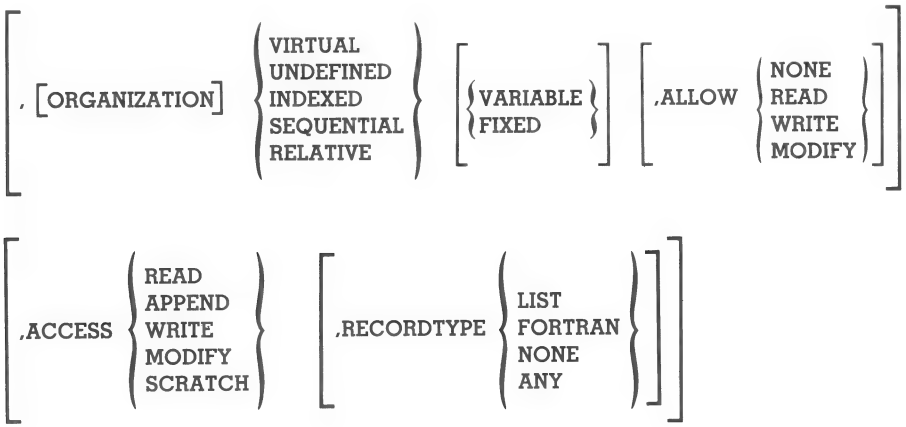
{ON ERROR / ONERROR} {GO TO 0 / GOTO 0}

Disables user error handling and returns to system error handling.

ON int-exp GOSUB lin-num [,lin-num]...
Transfers control to one of several subroutines, depending on the value of int-exp.

ON int-exp GOTO lin-num [,lin-num]...
Transfers control to one of several line numbers, depending on the value of int-exp.

OPEN str-exp1 [{FOR OUTPUT / FOR INPUT}] AS FILE #chnl-exp



OPEN (Cont.)

[,RECORDSIZE int-exp1]

[,FILESIZE int-exp2]

[,EXTENDSIZE int-exp3]

[,WINDOWSIZE int-exp4]

[,USEROPEN sub-nam]

[,TEMPORARY]

[,CONTIGUOUS]

[,DEFAULTNAME str-exp2]

[,MAP map-nam]

SEQUENTIAL files: **[,BLOCKSIZE int-exp7]**

[,NOREWIND]

[,NOSPAN]

RELATIVE files: **[,BUCKETSIZE int-exp8]**

[,BUFFER int-exp9]

INDEXED files: **[,BUCKETSIZE int-exp10]**

[,BUFFER int-exp11]

[,PRIMARY [KEY] vbl1 [DUPLICATES]

[,ALTERNATE [KEY] vbl2 [DUPLICATES] [CHANGES]

[,CONNECT chnl-exp2]

Enables access to a file or device.

PRINT [#chnl-exp] $\left[\left\{\begin{matrix},\\;\end{matrix}\right\}\right.$ **[exp]** $\left[\left\{\begin{matrix},\\;\end{matrix}\right\}\right.$ **[exp]...** $\left.\right]\left.\right]$

Transfers program data to a terminal or to a terminal-format file.

PRINT [#chnl-exp] USING str-exp $\left[\left\{\begin{matrix},\\;\end{matrix}\right\}\right.$ **[exp]** $\left[\left\{\begin{matrix},\\;\end{matrix}\right\}\right.$ **[exp]...** $\left.\right]\left.\right]$

Generates output formatted according to a format string for either numeric or string output.

PUT #chnl-exp [,COUNT int-exp]
Writes the next logical record from a record buffer to a sequential, relative, or indexed file.

PUT #chnl-exp, RECORD int-exp [,COUNT int-exp]
Writes the specified record from a record buffer to a relative or virtual file.

RANDOM[IZE]
Gives the random number function, RND, a new starting point.

READ vbl [,vbl]...
Assigns values from a DATA statement to each variable.

REM [comment]
Allows comments in a source program.

RESTORE [#chnl-exp [,KEY #int-exp]]
RESET
Resets the DATA statement pointer to the beginning of the DATA sequence, or sets the record pointer to the first record in the file.

RESUME [lin-num]
Ends an error handling routine and returns control to lin-num.

RETURN
Transfers control to the statement following the last ON GOSUB or GOSUB statement executed.

RSET str-vbl [,str-vbl ...] = str-exp
Assigns right-justified data to each str-var (does not change str-var length).

SCRATCH #chnl-num
Truncates a sequential file at the current record pointer.

SLEEP int-exp
Suspends program execution for int-exp seconds.

STOP
Halts program execution.

SUB sub-nam [([d-type] { variable / array [BY REF] } [,...])]

Begins a BASIC subprogram and specifies its parameters by number and data type.

SUBEND
Ends a BASIC subprogram and returns control to the calling program.

SUBEXIT
Returns control to the calling program. Equivalent to an unconditional branch to SUBEND.

stmnt UNLESS cond-exp
Executes the preceding statement only if cond-exp is false.

UNLOCK #chnl-exp
Unlocks the record or bucket accessed by the last FIND or GET.

UNTIL cond-exp
Begins an UNTIL loop.

stmnt UNTIL cond-exp
Executes the preceding statement repeatedly until cond-exp is true.

UPDATE #chnl-exp [,COUNT int-exp]
Replaces a record in a file with the record in the record buffer.

WAIT int-exp
Specifies the number of seconds the program waits for terminal input.

WHILE cond-exp
Begins a WHILE loop.

stmnt WHILE cond-exp
Executes the preceding statement repeatedly until cond-exp is false.

Functions

Result: I – integer, R – real number, S – string, N – no return value

Format	Result	Purpose
ABS(num-exp)	R	Provides the absolute value of num-exp.
ASCII(str-exp)	I	Provides the decimal ASCII value of the string's first character.
ATN(num-exp)	R	Provides the angle, in radians, of a specified tangent.
CCPOS(chnl-exp)	I	Provides the current character or cursor position on a channel.
CHR$(int-exp)	S	Provides a one-character string with the specified ASCII value.
COMP%(num-str1, num-str2)	I	Compares two numeric strings and returns +1 if num-str1 › num-str2, 0 if they are equal, or –1 if num-str1 ‹ num-str2.
COS(num-exp)	R	Provides the cosine of an angle's radian measure.
CTRLC	I	Enables CTRL/C trapping. Always returns 0.
DATE$(0%)	S	Provides the current date.
DET	R	Provides the determinant of the last matrix inverted by MAT INV.
DIF$(str-exp1, str-exp2)	S	Provides a string whose value is the difference between two numeric strings.
ECHO(chnl-exp)	I	Enables keyboard character echo. Returns 0.
EDIT$(str-exp, int-exp)	S	Performs editing operations, based on the additive value of int-exp: 1 – clears bit 0 of each character 2 – discards all spaces and tabs 4 – discards carriage returns, line feeds, form feeds, deletes, nulls 8 – discards leading spaces and tabs 16 – converts multiple spaces and tabs to a single space 32 – converts lowercase letters to uppercase letters 64 – converts brackets to parentheses 128 – discards trailing spaces and tabs 256 – suppresses editing of characters within quotation marks

Format	Result	Purpose
ERL	I	Provides the line number executing the last error.
ERN$	S	Provides the name of the program, DEF, or function subprogram executing when the last error occurred.
ERR	I	Provides the number of the last error.
ERT$(int-exp)	S	Provides explanatory text for a specified error number.
EXP(num-exp)	R	Provides the value of the mathematical constant "e" raised to a power.
FIX(num-exp)	I	Truncates a real number to an integer.
FORMAT$(num-exp, str-exp)	S	Converts a numeric value to a string, formatted according to str-exp using PRINT USING format rules.
INSTR(int-exp, str-exp1, str-exp2)	I	Searches for str-exp2 within str-exp1 and returns its starting position.
INT(num-exp)	R	Provides the largest whole real number less than or equal to num-exp.
LEFT[$](str-exp, int-exp)	S	Extracts a substring from a string's left side, int-exp characters long.
LEN(str-exp)	I	Provides a string's length.
LOC(symb)	I	Provides a longword virtual address of a variable or symbol.
LOG(num-exp)	R	Provides a number's natural log.
LOG10(num-exp)	R	Provides a number's common log.
MAR(chnl-exp)	I	Provides a channel's right margin.
MID[$](str-exp, int-exp1, int-exp2)	S	Extracts a substring int-exp2 characters long from str-exp, starting at int-exp1.
NOECHO(chnl-exp)	I	Disables keyboard character echo. Returns 0.
NUM	I	Provides the row number of the last element moved into an array by a MAT I/O statement.
NUM2	I	Provides the column number of the last element moved into an array by a MAT I/O statement.

Format	Result	Purpose
NUM$(num-exp)	S	Returns num-exp in PRINT statement numeric string format, possibly using rounding, suppression of trailing zeros and the decimal point, and E notation. Either a space or a minus sign occupies the first position; a space always occupies the last position. (See also NUM1$, STR$.)
NUM1$(num-exp)	S	Returns num-exp as a numeric character string without leading or trailing spaces. Unlike NUM$, NUM1$ neither uses E notation nor eliminates trailing zeros or the decimal point. (See also NUM$, STR$.)
PLACE$(num-str, int-exp)	S	Changes the precision of num-str, with either rounding or truncation.
POS(str-exp1, str-exp2, int-exp)	I	Provides str-exp2's starting character position within str-exp1, searching from int-exp.
PROD$(num-str1, num-str2, int-exp)	S	Provides the product of two numeric strings, with a precision of int-exp.
QUO$(num-str1, num-str2, int-exp)	S	Provides the quotient of two numeric strings, with a precision of int-exp.

RCTRLC	I	Disables CTRL/C trapping. Always returns 0.
RCTRLO (chnl-exp)	I	Cancels the effect of CTRL/O. Always returns 0.
RECOUNT	I	Provides the number of characters read by the last input operation.
RIGHT[$](str-exp, int-exp)	S	Extracts a substring from a string's right side, of int-exp characters.
RND	R	Provides a random number 0 <= n < 1.
SEG$(str-exp, int-exp1, int-exp2)	S	Extracts a substring from str-exp, starting at int-exp1 and ending at int-exp2.
SGN(num-exp)	I	Returns a –1, 0, or +1 if num-exp is negative, zero, or positive.
SIN(num-exp)	R	Returns the sine of an angle's radian measure.
SPACE$(int-exp)	S	Returns a string of int-exp spaces.
SQR(num-exp)	R	Returns a number's square root.

Format	Result	Purpose
STATUS	I	Provides information about the last channel opened: BIT SET 0 – record-oriented device 1 – carriage-oriented device 2 – terminal 3 – directory device 4 – single directory device 5 – sequential block-oriented device
STRING$(int-exp1, int-exp2)	S	Provides a string of length int-exp1 containing the same ASCII character (int-exp2) in each position.
STR$(num-exp)	S	Returns a numeric string like the NUM$ function, but without leading or trailing spaces.
SUM$(str-exp1, str-exp2)	S	Provides the sum of two numeric strings.
SWAP%(int-exp)	I	Transposes an integer's two low-order bytes.
TAB(int-exp)	N	Within a PRINT statement, moves the cursor or print mechanism rightward to the int-exp column.
TAN(num-exp)	R	Provides the tangent of an angle's radian measure.
TIME(num-exp)	R	Provides time information, depending on the value of num-exp: 0 – the time of day in seconds, relative to midnight 1 – the current job's CPU time in tenths of seconds 2 – the current job's connect time in minutes
TIME$(int-exp)	S	Returns the time of day in the form HH:MM PM or HH:MM AM.
TRM$(str-exp)	S	Returns a string identical to str-exp with trailing blanks and tabs removed. Does not alter str-exp.
VAL(num-str)	R	Provides a numeric string's floating point value.
VAL%(int-str)	I	Provides a numeric string's integer value.
XLATE(str-exp1, str-exp2)	S	Translates str-exp1 to str-var using a translation table (str-exp2).

Operators

Numeric Operators	+	**String Operators**	+	**Relational Operators**	=	**Logical Operators**	**NOT**
	−				‹		**AND**
	*				›		**OR**
	/				‹=, =‹		**XOR**
	^ **				›=, =›		**EQV**
					‹›, ›‹		**IMP**
					= =		

Operator Precedence

** or ^	Highest
– (unary minus)	
* or /	
+ or –	
+ (concatenation)	
all relational operators	
NOT	
AND	
OR, XOR	
IMP	
EQV	Lowest

Sample Program Coding Template

This section presents a format for coding BASIC programs. The recommended program order and documenting procedures clarify the program's history, purpose, and logical development.

```
1       !*****************************************************************************
        !
        !
        !        S U B - P R O G R A M   T I T L E
        !
        !
            SUB XXXXXX              ! Start sub-program here
        !   \             ON ERROR GOTO 19000
        !
        !
        !       Version:          VV
        !       Edit level:       EE
        !       Edit date:        DD-MMM-YY
        !       Author(s):        AAA
        !
        !*****************************************************************************

11      !*****************************************************************************
        !
        !                        C O P Y R I G H T
        !
        !   (C) Copyright 1982
        !   Digital Equipment Corporation, Maynard, Massachusetts
        !
        !   This  software  is  furnished  under  a  license  for  use  only
        !   on  a  single  computer  system  and  may  be  copied  only  with
        !   the  inclusion  of  the  above  copyright  notice.  This
        !   software,  or  any  other  copies  thereof,  may  not  be  pro-
        !   vided  or  otherwise  made  available  to  any  other  person
        !   except  for  use  on  such  system  and  to  one  who  agrees  to
        !   these  license  terms.  Title  to  and  ownership  of  the
        !   software shall at all times remain in DIGITAL.
        !
        !   The  information  in  this  software  is  subject  to  change
        !   without  notice  and  should  not  be  construed  as  a  commit-
        !   ment by Digital Equipment Corporation.
        !
        !   DIGITAL  assumes  no  responsibility  for  the  use  or  relia-
        !   bility  of  its  software  on  equipment  that  is  not  sup-
        !   plied by DIGITAL.
        !
        !*****************************************************************************
```

```
15  !*****************************************************************************
    !
    !                          C A L L   F O R M A T
    !
    !         CALL XXXXXX(<argument list, if present>)
    !
    !         Arguments:
    !         Name                        Description
    !         --------------------    ------------------------------
20  !*****************************************************************************
    !
    !         M O D I F I C A T I O N   H I S T O R Y   L O G
    !
    !         VER/ED   DATE   INITIAL REASON
    !         -------  -----  -------------------------------------
    !*****************************************************************************

100 !*****************************************************************************
    !         S U B - P R O G R A M   D E S C R I P T I O N
    !
300 !*****************************************************************************
    !         C H A N N E L     A S S I G N M E N T S
    !
    !         CHANNEL  ASSIGNMENT
    !
    !         -------- ------------------------------------
400 !*****************************************************************************
    !
    !         V A R I A B L E S   A N D   A R R A Y S   U S E D
    !         NAME                  DESCRIPTION
    !
    !         -----                 ------------------------------
    !
    !         Lines 400-449 contain local variables and arrays.
    !         Lines 450-499 contain APPENDed variables and arrays.
    !
    !*****************************************************************************
```

```
500  !*******************************************************************************
     !
     !       E X T E R N A L   S U B S   A N D   F U N C T I O N S
     !
     !       NAME                  DESCRIPTION
     !
     !       -----                 -------------------------------
     !
     !
700  !*******************************************************************************
     !
     !            S U B R O U T I N E S   U S E D
     !
     !       NAME/LINE       DESCRIPTION
     !
     !       ----------       ------------------------------------
     !
     !*******************************************************************************

750  !*******************************************************************************
     !
     !                    F U N C T I O N S   U S E D
     !
     !       LINE    NAME             DESCRIPTION
     !
     !       -----   -------------  ------------------------------
     !
800  !*******************************************************************************
     !
     !       C O M M O N   D E C L A R A T I O N S
     !
     !       Lines 800-849 contain local COMMON/MAP declarations.
     !       Lines 850-899 contain library COMMON/MAP declarations.
     !
     !*******************************************************************************
```

```
900     !*******************************************************************************
        !
        !       D I M E N S I O N   D E C L A R A T I O N S
        !
        !       Lines 900-949 denote local dimension declarations.
        !       Lines 950-999 denote library dimension declarations.
1000    !*******************************************************************************
        !
        !       M A I N   S U B - P R O G R A M   L O G I C
        !
10000   !*******************************************************************************
        !
        !       S U B R O U T I N E S   L O C A L   TO
        !
        !              T H I S   S U B - P R O G R A M
        !
        !*******************************************************************************
15000   !*******************************************************************************
        !
        !       F U N C T I O N S   L O C A L   TO
        !
        !              T H I S   S U B - P R O G R A M
        !
19000   !*******************************************************************************
        !
        !       S T A N D A R D   E R R O R   H A N D L I N G
        !
        !*******************************************************************************
19900           RESUME 32767
        !
32000   !*******************************************************************************
        !
        !       E N D   O F   P R O C E S S I N G
        !
32767   !*******************************************************************************
        !
        !       E N D   O F   S U B - P R O G R A M
        !
        !*******************************************************************************
        \ SUBEND
```

Principles of Transportability

Use correct style

- Declare all variables before use. Use DIM, COM, and MAP. For example:

```
100 DIM A%(100%)
110 C%=A%(50%)
```

- Dimension virtual arrays before OPEN. For example:

```
100 DIM #1%, ARRAY1%(100%)
200 OPEN "TDBRN.VIR" AS FILE #1%, &
    ORGANIZATION VIRTUAL
```

- Put only one statement on each text line. For example:

```
500 A$="DEVICE NAME"
600 B$="USER ID"
```

- Use different variable names for a scalar and an array. For example:

```
100 B%=C%(C.1%)
```

- To simplify error handling, put only one I/O or CALL per line. For example:

```
100 GET #1%
200 PUT #1%
```

- Place specialized system features in subprograms or functions with generalized parameters. These features include calls to system services.

Avoid side effects

- Explicitly close file I/O channels. For example:

```
32000 CLOSE #2%, #1%
```

- Do not jump into or out of a DEF.
- End error handlers with RESUME. For example:

```
5 ON ERROR GOTO 19000
    .
    .
    .
19000
19010 RESUME 32767
32767 END
```

- Do not rely on error variables outside the error handler. These variables are ERR, ERL, and ERN$. However, the error variable ERT$ is used anywhere.

Avoid default values

- Spell out all options on OPEN statements. For example:

```
10 OPEN "TEST.DAT" FOR INPUT AS FILE #1%, &
        ORGANIZATION SEQUENTIAL VARIABLE, &
        ALLOW READ, ACCESS MODIFY, MAP CTP
```

- Use integers for integer system functions. For example:

```
19010 IF ERR = 6% THEN . . .
```

Use recommended features

- Use POS for substring match, not INSTR.
- Use EDIT$ for editing strings, not CVT$$.
- Use MAP or MOVE, not CVT$$, FIELD, and so forth.
- Use COMPILE/WORD/NOOVERFLOW for PDP–11 compatibility.

Note

For more information on transporting BASIC programs, refer to the *BASIC Transportability Manual*, order number AA–L011A–TK.

ASCII Character Codes

DECIMAL CODE	8–BIT HEX CODE	CHARACTER	REMARKS
0	00	NUL	Null, tape feed
1	01	SOH	Start of heading, start of message, ^A
2	02	STX	Start of text, end of address, ^B
3	03	ETX	End of text, end of message, ^C
4	04	EOT	End of transmission, shuts off TWX machine, ^D
5	05	ENQ	Enquiry, WRU, ^E
6	06	ACK	Acknowledge, RU, ^F
7	07	BEL	Bell, ^G
8	08	BS	Backspace, ^H
9	09	HT	Horizontal tabulation, ^I
10	0A	LF	Line feed, ^J
11	0B	VT	Vertical tabulation, ^K
12	0C	FF	Form feed, page, ^L
13	0D	CR	Carriage return, ^M
14	0E	SO	Shift out, ^N
15	0F	SI	Shift in, ^O
16	10	DLE	Data link escape, ^P
17	11	DC1	Device control 1, ^Q
18	12	DC2	Device control 2, ^R
19	13	DC3	Device control 3, ^S
20	14	DC4	Device control 4, ^T

DECIMAL CODE	8-BIT HEX CODE	CHARACTER	REMARKS
21	15	NAK	Negative acknowledge, ERR, ^U
22	16	SYN	Synchronous idle, ^V
23	17	ETB	End-of-transmission block, logical end of medium, ^W
24	18	CAN	Cancel, ^X
25	19	EM	End of medium, ^Y
26	1A	SUB	Substitute, ^Z
27	1B	ESC	Escape, prefix
28	1C	FS	File separator
29	1D	GS	Group separator
30	1E	RS	Record separator
31	1F	US	Unit separator
32	20	SP	Space
33	21	!	Exclamation point
34	22	"	Double quotation mark
35	23	#	Number sign
36	24	$	Dollar sign
37	25	%	Percent sign
38	26	&	Ampersand
39	27	'	Apostrophe
40	28	(	Left parenthesis
41	29	)	Right parenthesis
42	2A	*	Asterisk
43	2B	+	Plus sign

DECIMAL CODE	8–BIT HEX CODE	CHARACTER	REMARKS
44	2C	,	Comma
45	2D	-	Minus sign, hyphen
46	2E	.	Period, decimal point
47	2F	/	Slash, slant
48	30	0	Zero
49	31	1	One
50	32	2	Two
51	33	3	Three
52	34	4	Four
53	35	5	Five
54	36	6	Six
55	37	7	Seven
56	38	8	Eight
57	39	9	Nine
58	3A	:	Colon
59	3B	;	Semicolon
60	3C	<	Less than, left angle bracket
61	3D	=	Equal sign
62	3E	>	Greater than, right angle bracket
63	3F	?	Question mark
64	40	@	Commercial at
65	41	A	Uppercase A
66	42	B	Uppercase B

DECIMAL CODE	8-BIT HEX CODE	CHARACTER	REMARKS
67	43	C	Uppercase C
68	44	D	Uppercase D
69	45	E	Uppercase E
70	46	F	Uppercase F
71	47	G	Uppercase G
72	48	H	Uppercase H
73	49	I	Uppercase I
74	4A	J	Uppercase J
75	4B	K	Uppercase K
76	4C	L	Uppercase L
77	4D	M	Uppercase M
78	4E	N	Uppercase N
79	4F	O	Uppercase O
80	50	P	Uppercase P
81	51	Q	Uppercase Q
82	52	R	Uppercase R
83	53	S	Uppercase S
84	54	T	Uppercase T
85	55	U	Uppercase U
86	56	V	Uppercase V
87	57	W	Uppercase W
88	58	X	Uppercase X
89	59	Y	Uppercase Y

DECIMAL CODE	8-BIT HEX CODE	CHARACTER	REMARKS
90	5A	Z	Uppercase Z
91	5B	[	Left square bracket
92	5C	\	Backslash, reverse slant
93	5D	]	Right square bracket
94	5E	^	Upward arrow head, caret, circumflex
95	5F	_	Underscore, underline
96	60	`	Grave accent
97	61	a	Lowercase a
98	62	b	Lowercase b
99	63	c	Lowercase c
100	64	d	Lowercase d
101	65	e	Lowercase e
102	66	f	Lowercase f
103	67	g	Lowercase g
104	68	h	Lowercase h
105	69	i	Lowercase i
106	6A	j	Lowercase j
107	6B	k	Lowercase k
108	6C	l	Lowercase l
109	6D	m	Lowercase m
110	6E	n	Lowercase n
111	6F	o	Lowercase o
112	70	p	Lowercase p

DECIMAL CODE	8-BIT HEX CODE	CHARACTER	REMARKS
113	71	q	Lowercase q
114	72	r	Lowercase r
115	73	s	Lowercase s
116	74	t	Lowercase t
117	75	u	Lowercase u
118	76	v	Lowercase v
119	77	w	Lowercase w
120	78	x	Lowercase x
121	79	y	Lowercase y
122	7A	z	Lowercase z
123	7B	{	Left brace
124	7C	|	Vertical line
125	7D	}	Right brace
126	7E	~	Tilde
127	7F	DEL	Delete, rubout

Predefined Constants

Constant	ASCII Value	Purpose
BEL (Bell)	7	Sounds the terminal bell.
BS (Backspace)	8	Moves the cursor one position to the left.
HT (Horizontal Tab)	9	Moves the cursor to the next horizontal tab stop.
LF (Line Feed)	10	Moves the cursor to the next line.
VT (Vertical Tab)	11	Moves the cursor to the next vertical tab stop.
FF (Form Feed)	12	Moves the cursor to the start of the next page.
CR (Carriage Return)	13	Moves the cursor to the beginning of the current line.
SO (Shift Out)	14	Shift Out (for alternate graphics).
SI (Shift In)	15	Shift In (for alternate graphics).
ESC (Escape)	27	Used to form escape sequences.
SP (Space)	32	Inserts one blank space in program output.
DEL (Delete)	127	Deletes the last character entered.
PI	None	Is the number PI (3.14159 in single precision).

VAX–11 BASIC Reserved Words

ABORT
ABS
ABS%
ACCESS
ACCESS%
ALIGNED
ALL
ALLOW
ALTERNATE
AND
ANY
APPEND
AS
ASC
ASCII
ATN
ATN2
BACK
BEL
BIN$
BINARY
BIT
BLOCK
BLOCKSIZE
BROADCAST
BS
BUCKETSIZE
BUFFER
BUFFERSIZE
BUFSIZ
BY
CALL
CALLR
CCPOS
CHAIN
CHANGE
CHANGES
CHR$
CLK$
CLOSE
CLUSTERSIZE
COM
COMMON
COMP%
CON
CONNECT
CONSTANT
CONTIGUOUS
COS
COT
COUNT
CR
CTRLC
CVT$$
CVT$%
CVT$F
CVT%$
CVTF$
DAT
DAT$
DATA
DATE$
DECLARE
DEF
DEF*
DEFAULTNAME
DEL
DELETE
DELIMIT
DENSITY
DESC
DET
DIF$
DIM
DIMENSION
DOUBLE
DOUBLEBUF
DUPLICATES
ECHO
EDIT$
ELSE
END
ENDIF
EQ
EQV
ERL
ERN$
ERR
ERROR
ERT$
ESC
EXP
EXTEND
EXTENDSIZE
EXTERNAL
FF
FIELD
FILE
FILESIZE
FILL
FILL$
FILL%
FIND
FIX
FIXED
FLUSH
FNEND
FNEXIT
FOR
FORCEIN
FORMAT$
FORTRAN
FREE
FROM
FSP$
FSS$
FUNCTION
FUNCTIONEND
FUNCTIONEXIT
GE
GET
GO
GOSUB
GOTO
GT
HANGUP
HEX
HEX$

Reserved Words (Cont.)

HT
IDN
IF
IFEND
IFMORE
IMAGE
IMP
INDEXED
INIMAGE
INPUT
INSTR
INT
INTEGER
INV
INVALID
JSB
KEY
KILL
LEFT
LEFT$
LEN
LET
LF
LINE
LINO
LINPUT
LIST
LOC
LOCK
LOF
LOG
LOG10
LONG
LSA
LSET
MAGTAPE
MAP
MAR
MAR%
MARGIN
MAT
MAX
MID
MID$
MIN
MOD
MOD%
MODE
MODIFY
MOVE
MSGMAP
NAME
NEXT
NOCHANGES
NODATA
NODUPLICATES
NOECHO
NOEXTEND
NOMARGIN
NONE
NOPAGE
NOQUOTE
NOREWIND
NOSPAN
NOT
NOTAPE
NUL$
NUM
NUM$
NUM1$
NUM2
OCT$
ON
ONECHR
ONENDFILE
ONERROR
OPEN
OR
ORGANIZATION
OUTPUT
PAGE
PEEK
PI
PLACE$
POKE
POS
POS%
PPS%
PRIMARY
PRINT
PRN
PROD$
PUT
QUO$
QUOTE
RAD%
RAD$
RANDOM
RANDOMIZE
RCTRLC
RCTRLO
READ
REAL
RECORD
RECORDATTR
RECORDSIZE
RECORDTYPE
RECOUNT
REF
RELATIVE
REM
RESET
RESTORE
RESUME
RETURN
RIGHT
RIGHT$
RND
RSET
SCRATCH
SEG$
SEQUENTIAL
SGN
SHIFT
SI
SIN
SINGLE
SLEEP
SO
SP
SPACE$
SPAN
SPEC%
SQR
SQRT
STATUS
STEP
STOP
STR$
STREAM
STRING
STRING$
SUB
SUBEND

Reserved Words (Cont.)

SUBEXIT
SUBROUTINE
SUM$
SWAP%
SYS
TAB
TAN
TAPE
TASK
TEMPORARY
TERMINAL
THEN
TIM
TIME
TIME$
TO
TRM$
TRN
TST
TSTEND
TYP
TYPE
TYPE$
UNALIGNED
UNDEFINED
UNLESS
UNLOCK
UNTIL
UPDATE
USEAGE
USEAGE$
USEROPEN
USING
USR
USR$
VAL
VAL%
VALUE
VARIABLE
VFC
VIRTUAL
VPS%
VT
WAIT
WHILE
WINDOWSIZE
WITH
WORD
WRITE
WRKMAP
XLATE
XOR
ZER